This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

Claim 1 (Currently Amended): A method of generating and storing a list of safe exception handlers ~~validating and dispatching an event~~, comprising:

identifying valid exception handlers;

including a safe exception section for compiled objects, said safe exception section comprising addresses of valid exception handlers in the section;

marking the compiled object if the compiled object references or contains at least one valid exception handler, said marking operable to identify that the object is associated with a valid exception handler;

generating at least one [[a]] list of valid exception handlers, said at least one list comprising the valid exception handlers included or referenced in the compiled objects[[,]] and storing said at least one list of valid exception handlers ~~located~~ in a protected area during program execution[[;]]

~~receiving an event;~~

~~determining an exception handler for the event;~~

~~determining if the exception handler is valid by comparing the exception handler to said list of valid exception handlers and determining if the exception handler is unaltered; and otherwise determining that the exception handler is invalid; and~~

~~executing the exception handler if the exception handler is valid~~.

Claim 2 (Canceled).

Claim 3 (Previously Presented): The method of claim 1, further comprising one of receiving and generating the list of valid exception handlers.

Claim 4 (Currently Amended): The method of 36 [[1]], further comprising retrieving a list of valid exception handlers from a storage device and comparing the exception handler to the list of valid exception handlers in determining if the exception handler is valid.

Claim 5 (Original): The method of claim 1, further comprising generating a list of valid exception handlers by compiling code into at least one of an object file and an image.

Claim 6 (Canceled)

Claim 7 (Currently Amended): The method of 36 [[1]], further comprising, if the exception handler is valid, determining whether the exception handler handles the event, and if so, executing the exception handler, and otherwise, retrieving a second exception handler from information on a stack and continuing processing with determining if the second exception handler is valid.

Claim 8 (Currently Amended): The method of 36 [[1]], further comprising terminating the method if the exception handler is invalid.

Claim 9 (Currently Amended): The method of claim 36 [[1]], further comprising generating an error message if the exception handler is invalid.

Claim 10 (Currently Amended): The method of claim 36 [[1]], further comprising, if the exception handler is valid, verifying other data for the event.

Claim 11 (Original): The method of claim 10, wherein the other data comprises pointer data.

Claim 12 (Currently Amended): A computer-readable storage medium having stored thereon computer-executable instructions for generating and storing a list of safe exception handlers

~~performing a method of validating and dispatching an event~~, the ~~method~~ medium comprising instructions for:

identifying valid exception handlers;

including a safe exception section for compiled objects, said safe exception section comprising addresses of valid exception handlers in the section;

marking the compiled object if the compiled object references or contains at least one valid exception handler, said marking operable to identify that the object is associated with a valid exception handler;

generating at least one [[a]] list of valid exception handlers, said at least one list comprising the valid exception handlers included or referenced in the compiled objects[[,]] and

storing said at least one list of valid exception handlers ~~located~~ in a protected area during program execution[[;]].

Claim 13 (Canceled)

Claim 14 (Previously Presented): The computer-readable storage medium of claim 12, having further computer-executable instructions for one of receiving and generating the list of valid exception handlers.

Claim 15 (Currently Amended): The computer-readable storage medium of claim 37 ~~12~~, having further computer-executable instructions for retrieving a list of valid exception handlers from a storage device and comparing the exception handler to the list of valid exception handlers in determining if the exception handler is valid.

Claim 16 (Previously Presented): The computer-readable storage medium of claim 12, having further computer-executable instructions for generating a list of valid exception handlers by compiling code into at least one of an object file and an image.

Claim 17 (Canceled).

Claim 18 (Currently Amended): The computer-readable storage medium of claim 37 ~~12~~, having further computer-executable instructions for, if the exception handler is valid, determining whether the exception handler handles the event, and if so, executing the exception handler, and otherwise, retrieving a second exception handler from information on a stack and continuing processing with determining if the second exception handler is valid.

Claim 19 (Currently Amended): The computer-readable storage medium of claim 37 ~~12~~, having further computer-executable instructions for terminating the method if the exception handler is invalid.

Claim 20 (Currently Amended): The computer-readable storage medium of claim 37 ~~12~~, having further computer-executable instructions for generating an error message if the exception handler is invalid.

Claim 21 (Currently Amended): The computer-readable storage medium of claim 37 ~~12~~, having further computer-executable instructions for, if the exception handler is valid, verifying other data for the event.

Claim 22 (Previously Presented): The computer-readable storage medium of claim 21, wherein the other data comprises pointer data.

Claim 23 (Currently Amended): A system for executing safe exceptions ~~validating an event to be dispatched~~, comprising:

a processor ~~that~~ configured to receive[[s]] an event; and

an exception dispatcher system ~~that~~ configured to determine an exception handler for the event, further comprising determining if the exception handler is valid by comparing the exception handler to a list of valid exception handlers and determining if the exception handler is unaltered, and otherwise determine that the exception handler is invalid, wherein said list of valid exception handlers are located in a protected area during program execution

and said list comprises exception handlers included or referenced in a safe exception section for compiled executable objects

~~generates a list of valid exception handlers, said list of valid exception handlers located in a protected area during program execution, determines an exception handler for the event, determines if the exception handler is valid by comparing the exception handler to said list of valid exception handlers and determining if the exception handler is unaltered, and otherwise determining that the exception handler is invalid; and executes the exception handler if the exception handler is valid~~.

Claim 24 (Canceled)

Claim 25 (Previously Presented): The system of claim 23, further comprising a storage device that stores a list of valid exception handlers, and the exception dispatcher system retrieves the list of valid exception handlers from the storage device and compares the exception handler to the list of valid exception handlers in determining if the exception handler is valid.

Claim 26 (Canceled).

Claim 27 (Canceled).

Claim 28 (Canceled).

Claim 29 (Previously Presented): The system of claim 23, wherein the exception dispatcher system, if the exception handler is valid, determines whether the exception handler handles the event, and if so, executes the exception handler, and otherwise, retrieves a second exception handler from information on a stack and continues processing with determining if the second exception handler is valid.

Claim 30 (Previously Presented): The system of claim 23, wherein the exception dispatcher system terminates processing if the exception handler is invalid.

Claim 31 (Previously Presented): The system of claim 23, wherein the exception dispatcher system generates an error message if the exception handler is invalid.

Claim 32 (Currently Amended): The system of claim 23, wherein an image is provided to the exception dispatcher system, said image created based on at least one object file received from at least one of a compiler and an assembler ~~further comprising a linker that creates an image based on at least one object file received from at least one of a compiler and an assembler, and provides the image to the exception dispatcher system~~.

Claim 33 (Canceled).

Claim 34 (Original): The system of claim 23, wherein the exception dispatcher system, if the exception handler is valid, verifies other data for the event.

Claim 35 (Original): The system of claim 34, wherein the other data comprises pointer data.

Claim 36 (New): The method of claim 1, further comprising:

receiving an event;

determining an exception handler for the event;

determining if the exception handler is valid by comparing the exception handler to said list of valid exception handlers and determining if the exception handler is unaltered; and otherwise determining that the exception handler is invalid; and

executing the exception handler if the exception handler is valid.

Claim 37 (New): The computer-readable storage medium of claim 12, further comprising instructions for:

receiving an event;

determining an exception handler for the event;

determining if the exception handler is valid by comparing the exception handler to said list of valid exception handlers and determining if the exception handler is unaltered; and otherwise determining that the exception handler is invalid; and

executing the exception handler if the exception handler is valid.